**Roulette Game**

Curtis Stephens

**Cis 17-c Midterm Project**

Table of Contents

Introduction3

Project Goals3

Game Play4

User Accounts4

Play by Color5

Play by Number5

Player Profile Options6

Technical End of Things

(What You Are Looking For)7

Classes7

Functions8

Containers9

Recursion9

Iterators10

Algorithms10

Room for Improvement (Project 2 Ideas)11

Introduction

Welcome to my roulette game! Roulette has always been one of my favorite casino games and when the project was proposed the very first game that came to mind was roulette. My favorite way to play is the video roulette machines many casinos are finding to be more popular. The ease of betting smaller amounts on more numbers is what appeals to these casinos. More games correlates with more money for the casinos. One thing that appealed to me about using roulette as my game for the project is that there is a winning number for every spin, creating a need for data storage. The greater the frequency of a number being hit, the more inclined the player feels to bet on that number. This game utilizes this inclination as well as other aspects of the game to create a seamless one on one gaming experience.

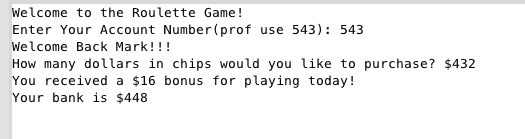
**Project goals**

Since this class is a data structures class, I wanted to choose a game that created a lot of data to sift through. The game plays flow, while is extremely user friendly, does have many side options you would not find in a normal video roulette machines. These options were added to manipulate the data to fulfill the requirements for this project. My goal with the project was to use iterators and algorithms to display and manipulate containers to the best of my knowledge. The STL library is a little much to take in in just a couple weeks and a lot of the research I did for this project were calls and concepts not presented in class but this project displays a working knowledge of the STL libraries.

Game Play

The game play in the game is broken down into two separate types of play. Though both play on the same style of wheel (a number generated between 0-36; with 36 representing the green 00), the odds in each type of game play vary drastically. Below is an overview of the options throughout the different menus.

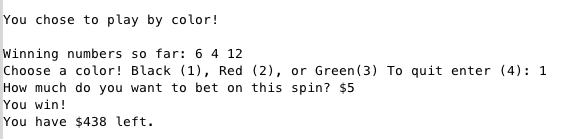
**User Accounts**



When the game opens up you are asked to enter your user ID number. For professor use an account number is provided, but if an ID is entered that is not valid the used is asked to set up a new account. A player’s account is then activated creating a new game. The user is then asked to purchase chips; which are stored in the user’s bank total. A bonus is also granted to any player who chose to try the new roulette game and that is then added to the bank total as well.

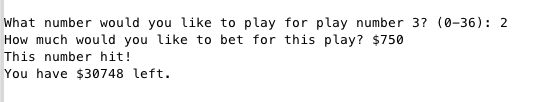
Now the game actually begins….

**Play by Color**

****

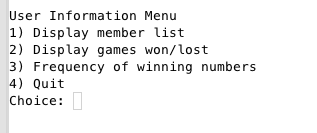
Playing by color is the obviously better odds in the game. A number is randomized and stored as the winning number for each spin. For this game playing red and black is the same as playing odd or even. Playing green pays out 18 to 1 if the random number is 0 or 36, while red or black pays out 2 to 1 if it hits between 1-35; with red being odd and black being even. Each spins results are recorded and stored for the player. Winning and losing numbers are displayed before each spin to inform the player what numbers might be “hot”. This process is repeated until the used chooses to quit playing this style of gaming.

**Play by Number**

****

Playing by number is not nearly as straight forward as playing by color, as anyone who as played roulette knows. For game play ease the user is only allowed to play up to 10 different numbers for each spin. The player is asked what numbers they would like to play and how much they would like to bet on each number. A number is then randomized and stored with each bet being compared to this winning number. Hitting on a winning number pays out 40 to 1. The players winning and losing numbers are stored in the player’s portfolio. This process is also repeated until the player chooses to quit.

**Player Profile Options**

****

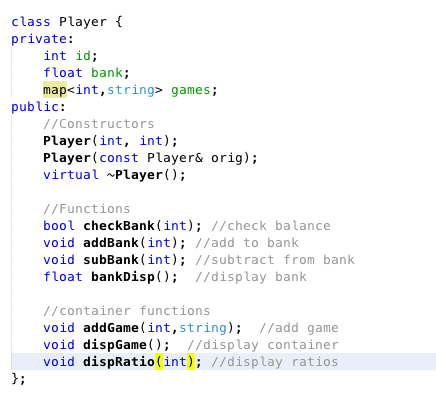
The player profile options menu is not something you would find on a normal game; mostly because the casinos do not want you to realize how bad you are doing. First option will display the member list; which I added to demonstrate the use of printing a map. Second option displays the frequency of winning numbers; which was added to demonstrate the use of iteration through a vector with multiple variables and counting the matches. Finally the third option prints the games won and lost by the player.

Technical End of Things

(What You Are Looking For)

While my code is not too far past the project requirement, it was much longer before some house cleaning. The try catch loops for input verification not only took up too much space; it also made the project less understandable. I passed all of the repetitious try-catch loops to appropriate functions based on constraints. More will be explained below.

**Classes**

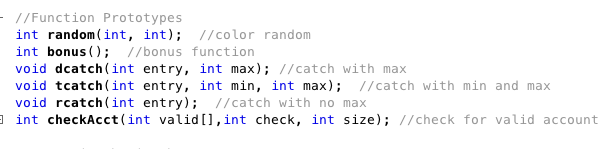


The class that needs little explanation is the NegChoice class. I used this to catch negative values entered into the program.

The class that does require explanation would be the Player class. This classes private members functions will include the user’s ID, as well as bankroll, and then a map consisting of the users number of games correlated to their result from each game.

The public members of this class consist of the constructors and functions. The constructors accept the ID and chip count to initialize while the destructor breaks down the map when the program ends. The functions are essentially setters and getters with the exception on a bank check and a map display

**Functions**

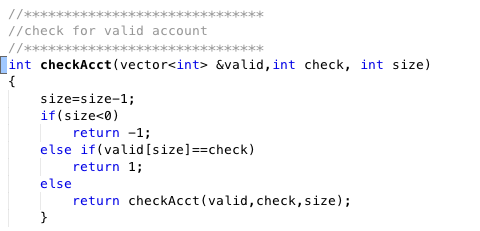
****

The functions used in this program were strictly for house keeping. Anything that is called more than once is passed off to a function. The main ones were the ones involving a try-catch. Going through for error checking with all the try catches involved cluttered the project. The last function listed was added as a search through recursion.

**Containers**

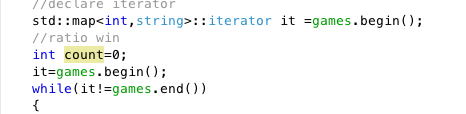
I used several containers throughout my project to demonstrate a working knowledge of the STL. The one utilized the most would be the map container. I used an integer-string mapping for both the accounts and the games win/loss features. Also used are vectors and queues, these are for tracking the winning and losing games and numbers.

**Recursion**

****

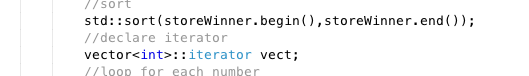
One concept I have spent I have found most intriguing so far would be recursion. To demonstrate recursion in my project I have passed the valid accounts vector through a recursive function.

**Iterators**

****

Throughout the entire project iterators are used for traversing the containers. The only for/while loops using normal counters are for game flow. Other than that all movement is done through the iterators.

**Algorithms**

****

Algorithms are essential to most container manipulations. The easiest way to check the frequency of a winning number is given in this example. Sorting my vector makes checking the frequency much easier. Without the sort counting duplicates would be almost impossible. Other examples can be found throughout the project including in the Player class.

Room For Improvement

(Project 2 Ideas)

One concept I struggled with that I ended up switching in my project due to time restrictions was the use of a multimap. I realized after doing my final builds I was compiling without C++ 11 enabled, which is the reason some of the function calls I was trying to use wouldn’t work. I spent many hours working out functions for certain features I was eventually able to solve through the STL. The biggest example of this would be my vector frequency count.

One idea I toyed with was to create a sort of “dirty” roulette table. Depending how much you tipped the attendant would increase your odds in different ways. Of course these odd increases would not be apparent to the user playing; but say they tipped over 20, then only numbers that were generated in the last 20 plays would be randomized. Since most players like to play off what numbers are hot, this drastically increases their chances of hitting. This, of course, would depend on what concepts we cover and need to incorporate by then.

Any ideas/comments/constructive criticism on what is here and what I can do to improve this project are gladly accepted. If there is any chance you can cover more on the STL library in class would be helpful. Most of what is here is through my own research. I know moving forward in computer science I will cross paths with these libraries.